CS 101 Lecture 26

Flow Control: Repetition with Loops
(Alice In Action, Ch 4)

Objectives

• Review using the \texttt{if} statement to perform some statements while skipping others
• Use the \texttt{for} and \texttt{while} statements to perform (other) statements more than once
Flow Control

- Flow: sequence of steps for performing a user story
- Flow control statement: structure for managing flow
- Flow control statements used in previous chapters
  - `doInOrder`: produces a sequential execution
  - `doTogether`: produces a parallel execution
  - methods and functions: name a block of statements

![Diagram showing `doInOrder` and `doTogether` statements](image)

Flow Control: if statement

- Control statements introduced in the current chapter
  - `if`: directs program flow along one of two paths

![Diagram showing flow through an `if` statement](image)
Flow Control: while loop

- Control statements introduced in the current chapter
  - while: directs flow into an arbitrary number of loops

---

Introducing Repetition

- Selection:
  - The if statement executes its body 0 or 1 times based on the condition.
- Indefinite Loop
  - The while loop executes its body 0 or more times based on the condition.
- Definite loop
  - The for loop executes its body a fixed number of times.
Introducing the **while** Statement

- Strategy for building third shot of dragon animation
  - Repeatedly have dragon flap its wings
  - Move dragon downward while it is above drawbridge
- Overview for building the third shot
  - Place `doTogether` statement in `doInOrder` statement
  - Use `setPointOfView()` to move camera closer
  - Send `flappingWings()` message to the **dragon**
  - Drag the **while** statement to the editing area
  - Drop the **while** statement below `doInOrder`
  - Insert placeholder value into the **while** condition

---

*FIGURE 4-26* An empty **while** statement
Introducing the **while** Statement (continued)

- Building the third shot (continued)
  - Drag dragon’s `isAbove()` over condition and drop
  - Add `castle.bridge` argument to `isAbove()`
  - Insert `doTogether` statement in the `while` loop
  - Add `move()` method to cause dragon to descend
  - Send another `flapWings()` message to `dragon`
  - Use `setPointOfView()` to zoom in for a close-up
- Infinite loop occurs if loop lacks falsifying condition
  - In third shot, `move()` eventually terminates loop
Introducing the **while** Statement (continued)

Comparing the **if** and **while** Statements

- Both test conditions before flow enters structure
- Both are bypassed if initial condition is **false**

- **while** statement is repeats after finishing its body.
  - 0 or more times.
- **if** statement’s body can only be executed once.
  - 0 or 1 times.
- **if** statement can have an **else** body.
Flow Control: for loop

- Control statements introduced in the current chapter
  - `while`: directs flow into an arbitrary number of loops

![Flow Control: for loop diagram](image)

Introducing Definite Loop

- Refer to `flapWings()` method from Figure 2-16
- Enhancement: use `for` loop to flap wings `numTimes`
- Overview for implementing the enhancement
  - Open the `flapWings()` method
  - Adjust the `duration` values for the wing movements
  - Drag `loop` control to the top of the method and drop
  - Select `numTimes` for number of iterations
  - Drag the `doInOrder` statement into the `for` statement
Introducing Repetition (continued)

FIGURE 4.17  Dropping the loop control

Introducing Repetition (continued)

FIGURE 4.19  The revised FlapWings() method
Mechanics of the **for** Statement

- Repeat statement execution a fixed number of times
- Example: pass 3 to `flapWings()` for 3 flaps
- Structure of the simple **for** statement
  - `for(int index = 0; index < limit; index++){
    Statements
  }
- The **for** statement is also known as a counting loop
  - First statement in () initializes the **index**
  - Second statement in () checks **index** against **limit**
  - Third statement in () increments the **index**

---

**Mechanics of the **for** Statement**  
(continued)

![Flow through a for statement](image_url)

**FIGURE 4-20** Flow through a **for** statement
Mechanics of the for Statement (continued)

• To test a for loop, trace the behavior with values
  – Statements are executed while \( \text{index} < \text{numTimes} \)
  – Example: send \( \text{flapWings}(3) \) to the \text{dragon} object
Mechanics of the for Statement (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Flow is in...</th>
<th>Effect</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>index &lt; numTimes (3 &lt; 3)</td>
<td>The condition is true</td>
<td>Flow is directed into the loop</td>
</tr>
<tr>
<td>9</td>
<td>doInOrder</td>
<td>Flap wings</td>
<td>The third repetition</td>
</tr>
<tr>
<td>10</td>
<td>index++</td>
<td>Increment index</td>
<td>index’s value changes from 2 to 3</td>
</tr>
<tr>
<td>11</td>
<td>index &lt; numTimes (3 &lt; 3)</td>
<td>The condition is false</td>
<td>Flow is directed out of the loop</td>
</tr>
<tr>
<td>12</td>
<td>Flow leaves the for statement, moving to the end of the method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 4-21 Tracing the flow of flapWings(3) (continued)

Mechanics of the for Statement (continued)

• Purpose of show complicated version button
  – Change initial value of index and/or update to index
  – Example: change update to index+=2

• Simple version of for lets you modify limit value
• Note: neither version of for allows you to count down
The **while** vs **for** loop

- **for loop property**: limit must be set to a fixed value
- Circumstance when the **for** loop is appropriate
  - Statements are to be executed a fixed number of times
  - “Defined” ahead of time → Definite loop.
- Problem: looping when the limit value is unknown
  - e.g. how many flaps for “Dragon then descends and lands on the drawbridge”?
  - Solution: use an **while** statement
  - Not “defined” ahead of time → Indefinite loop.

Comparing the **for** and **while** Statements

- **while** statement is **more general** and can produce any type of repetition, including the **for** loop behavior
- **for** statement is used for **fixed number of repetitions**
  - Loop selection question to ask: “Am I counting?”
  - If yes, use a **for** statement; otherwise, use **while**
- Both loops test conditions before flow enters structure
- Both loops are bypassed if initial condition is **false**
- More examples:
  - drop a ball: continue the bounce **while** rebound distanceToGround >0
  - FabonacciGirl: use **for** to compute Fabonacci numbers and move corresponding steps
A While-Loop Example

• Setting up the scene
  – Use shebuilder to create a soccer player (Jane)
  – Place soccerBall object in Jane’s hands
• Writing a dropBounce() method for soccerBall
  – Move the ball down distanceToGround meters
  – Change distanceToGround by bounce factor (2/3)
  – Move ball up reduced distanceToGround meters
  – Bouncing continues while distanceToGround > 0
• Writing a janeDropsBall() method
  – Send roll() messages to forearms
  – Send dropAndBounce() message to soccerBall

A While-Loop Example (continued)

![Diagram of bouncing ball motion]

FIGURE 4-31  Sketch of the up-down motion of a bouncing ball
A While-Loop Example (continued)

FIGURE 4.32 Jane with the soccer ball

A While-Loop Example (continued)

FIGURE 4.33 Method soccerBall.dropAndBounce()
Nested Loops

- Example: three shots enhancing Scene 1 of dragon animation
  - Dragon flies toward a castle in the countryside
  - As dragon nears castle, it circles the tower three times
  - Dragon then descends and lands on the drawbridge (section 4.4)

- One way to build the first shot
  - Go to go into the Add Objects window
  - Position the dragon above the castle’s drawbridge
  - Move dragon up until it is even with the castle’s tower
  - Drop a dummy and then drag the dragon off-screen
  - Use setPointOfView() to properly position dragon

Nested Loops (continued)

- One way to build the second shot: circling the tower 3 times
  - Outer for loop for circling the tower 3 times
  - inner for loop in flapWings() for flapping wings 4 times in each circle
  - More Alice details
    - AsSeenBy() attribute revolves dragon around castle
    - Increase duration of turn() to synchronize moves
    - Set style suitable for animation
Fibonacci Numbers

• The original problem investigated by Fibonacci in 1202
  – about how fast rabbits could breed in ideal circumstances

  – Suppose a newly-born pair of rabbits, one male, one female, are put in a field.
  – Rabbits are able to mate at the age of one month so that at the end of its second month a female can produce another pair of rabbits.
  – Suppose that our rabbits never die and that the female always produces one new pair (one male, one female) every month from the second month on.

  – How many pairs will there be in one year?

Fibonacci Numbers

• Analysis
  – At the end of the first month, they mate, but there is still one only 1 pair.
  – At the end of the second month the female produces a new pair, so now there are 2 pairs of rabbits in the field.
  – At the end of the third month, the original female produces a second pair, making 3 pairs in all in the field.
  – At the end of the fourth month, the original female has produced yet another new pair, the female born two months ago produces her first pair also, making 5 pairs.
  – …
Fibonacci Numbers in the Rabbit Growth Model

- Fibonacci numbers (sequence)
  - Number > 2 is found by adding two preceding numbers
  - Example: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

Fabonacci Numbers in Nature

http://britton.disted.camosun.bc.ca/fibslide/jbfibslide.htm
Fibonacci Number Example

- User story
  - Scene 1: girl finds an old book and reads contents
  - Scene 2: girl uses the map to locate a palm tree
  - Scene 3: girl follows spiral from tree to treasure site
- Coding spiral motion in Scene 3 is greatest challenge
  - Spiral inscribed in rectangles built from the sequence

http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html
Spirals and the Fibonacci Function (continued)

• Approximating Fibonacci spiral in `playScene3()`
  – Have girl move 6 times
  – Distance of each move equals a corresponding Fibonacci number
  – While moving forward, girl also turns left 1/4 revolution
• `playScene3()` requires a `fibonacci()` function, which will be defined as an object method for the girl
• Save girl as `fibonacciGirl` for possible reuse

![Figure 4-36: The `playScene3()` Method]
The Fibonacci Function

• Defining the outline of `fibonacci()` function
  – Select girl and create a function named `fibonacci`
  – Create a `Number` parameter named `n`

• Formula: if \( n > 1 \), \( f(n) = \) sum of two preceding numbers

• Designing an algorithm to generate the \( n^{th} \) number
  – Create local variables: `result`, `nextToLast`, `last`
  – Add if statement to the function
  – If \( n == 1 \) or \( n == 2 \), \( result = 1 \)
  – Otherwise calculate \( n^{th} \) value using formula in `for` loop

• `fibonacci()` calls in `playScene3()` specify spiral
Summary

- Flow control statement: controls the flow of statement execution
- **if** statement: directs flow along one of two paths based on evaluation of a condition
- **for** statement: repeats execution of a group of statements a fixed number of times
- **while** statement: repeats execution of a group of statements an appropriate number of times based on a loop continuation condition
- **Design Patterns** for writing programs.
  - Loops: Interactive, Counter, Sentinel
  - Accumulator Pattern

Announcements and To Do

- Readings:
  - Alice in Action, Chapter 4 (This week)

- Check schedule for HW assignments
Flow of Control

**Sequential Execution**
Each instruction is executed in order they are written (after the previous one, before the next one).

**Functions**
Enable procedural decomposition.
Repeat statements by calling functions multiple times.

---

Flow of Control

**Selection**
Some statements are executed while others are not.

**Repetition**
Statements can be repeated some fixed number of time, or else can be repeated until some event signals they should not be repeated any more.
Design Patterns for Indefinite Loops

Several common patterns:
• Counter-controlled Loop
• Interactive Loop
• Sentinel-controlled Loop

Also… accumulator design pattern

Counter-Controlled Pattern

Counter-Controlled Loop
Evaluates a counter in loop’s logical expression:

Pseudo-code for this pattern:
initialize counter
while counter > critical-value
do controlled block statements
decrement (increment) counter
Interactive Loop Pattern

Interactive Loop
Prompts user whether or not to continue.

Pseudo-code for this pattern:
set moredata to “yes”
while moredata is “yes”
  get next data item
  process data item
  ask user if there is moredata

Sentinel Loop Pattern

Sentinel Loop
Checks input for a special value to determine whether or not to continue.

eg. Use -1 while entering quiz grades means STOP.

get first data item
while data item is not the sentinel
  process data item
  get next data item
Accumulator Pattern

Accumulator Pattern
Used to combine many values into one value.
Eg. Sum several values in a loop. Can be used with any loop pattern.

initialize accumulator variable(s)
loop to repeat each item:
    combine new data item into accumulator

Indefinite Loop Problems

Infinite Loop
A loop with no end – no way out!
Generally, this is a very bad problem.

Example:
an argument with... your parents?
Have you ever noticed?

Indefinite Loop Problems

Busy Loop
An infinite loop, which never pauses (e.g. for input), and thus consumes all of the computer’s resources.
This is a terrible, awful, no-good, very bad problem.

Example:
a counting loop with a faulty condition.